**1. Hosting Static Website on Amazon S3**

(AWS S3 is best for static HTML/CSS/JS websites, no backend logic.)

**Step 1: Create an S3 Bucket**

* Go to AWS Console → S3 → Create bucket.
* Bucket name must be globally unique (e.g., mywebsite-demo).
* Choose region close to your users.

**Step 2: Upload Website Files**

* Upload your HTML, CSS, JS, images, etc.
* Keep your entry file named index.html.

**Step 3: Enable Static Website Hosting**

* In your bucket → Properties → "Static website hosting".
* Enable it, set **index document** to index.html.
* Copy the website endpoint given.

**Step 4: Set Permissions (Public Access)**

* Go to Permissions → Uncheck “Block all public access”.
* Add a **bucket policy** like:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::mywebsite-demo/\*"

}

]

}

**2. Hosting Static Website on Amazon EC2**

(EC2 gives you a VM where you install a web server like Apache or Nginx.)

**Step 1: Launch EC2 Instance**

* Go to EC2 → Launch Instance.
* Choose Amazon Linux 2 (or Ubuntu).
* Select instance type (t2.micro for free tier).
* Configure → Create new key pair (for SSH).
* Launch.

**Step 2: Connect to Instance**

* Use SSH from your terminal:

ssh -i my-key.pem ec2-user@<EC2-Public-IP>

**Step 3: Install Web Server**  
For Amazon Linux:

sudo yum update -y

sudo yum install -y httpd

sudo systemctl start httpd

sudo systemctl enable httpd

For Ubuntu:

sudo apt update

sudo apt install -y apache2

sudo systemctl start apache2

sudo systemctl enable apache2

**Step 4: Upload Website Files**

* Copy files via scp:

scp -i my-key.pem index.html ec2-user@<EC2-Public-IP>:/var/www/html/

* Place all HTML/CSS/JS files in /var/www/html/.

**Step 5: Security Group Rules**

* In EC2 → Security Groups → Inbound Rules.
* Add rule: HTTP → Port 80 → Source: Anywhere (0.0.0.0/0).

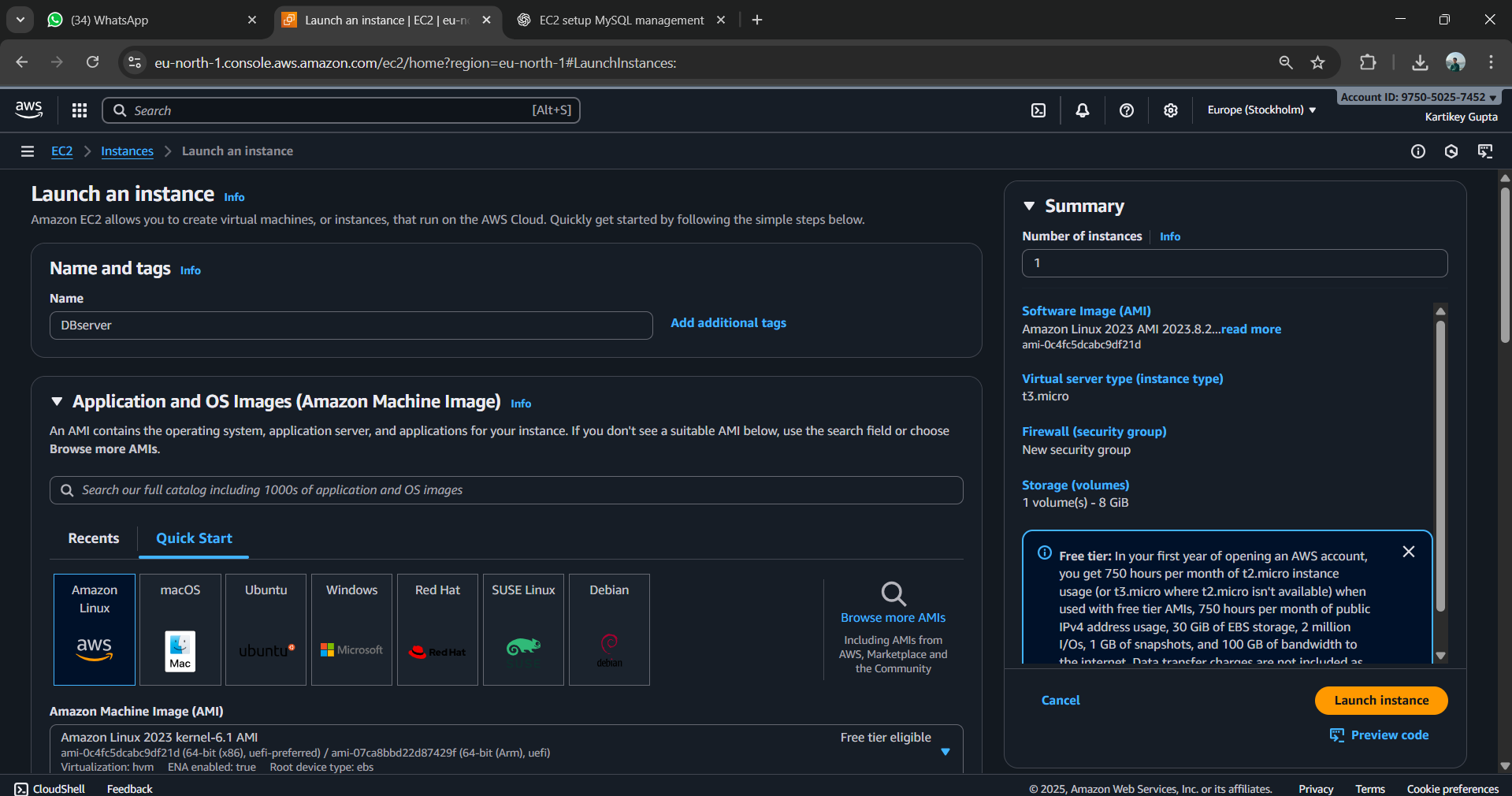
**Step 6: Access Website**

* Open http://<EC2-Public-IP> in browser.

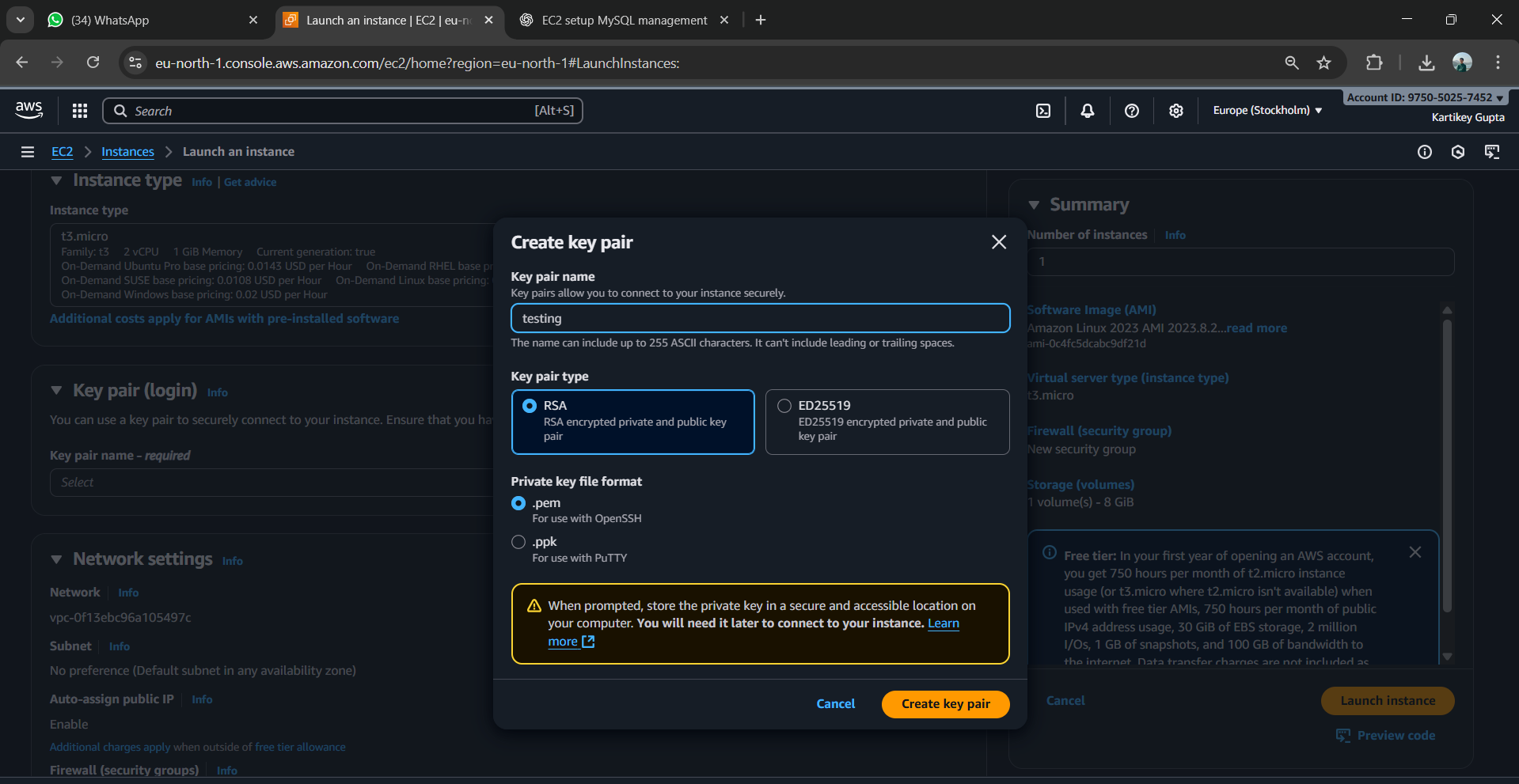
**SQL**

**1. Launch EC2 Instance**

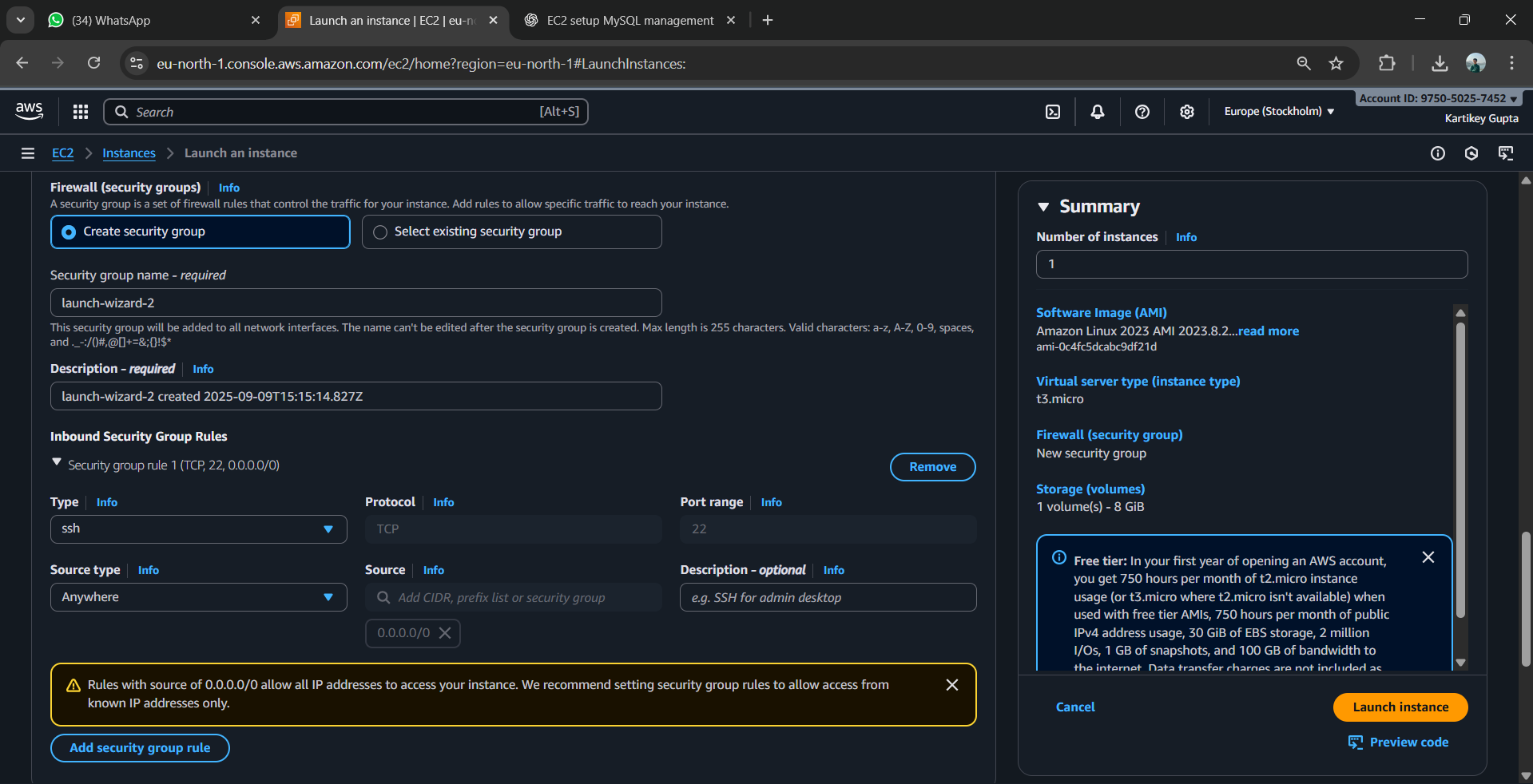
1. Open **AWS Console → EC2 → Launch Instance**.
2. Choose **Amazon Linux 2 AMI (Free Tier Eligible)**.
3. Select **t2.micro** (Free Tier).



1. Create/download a **Key Pair (.pem)** (needed for SSH).



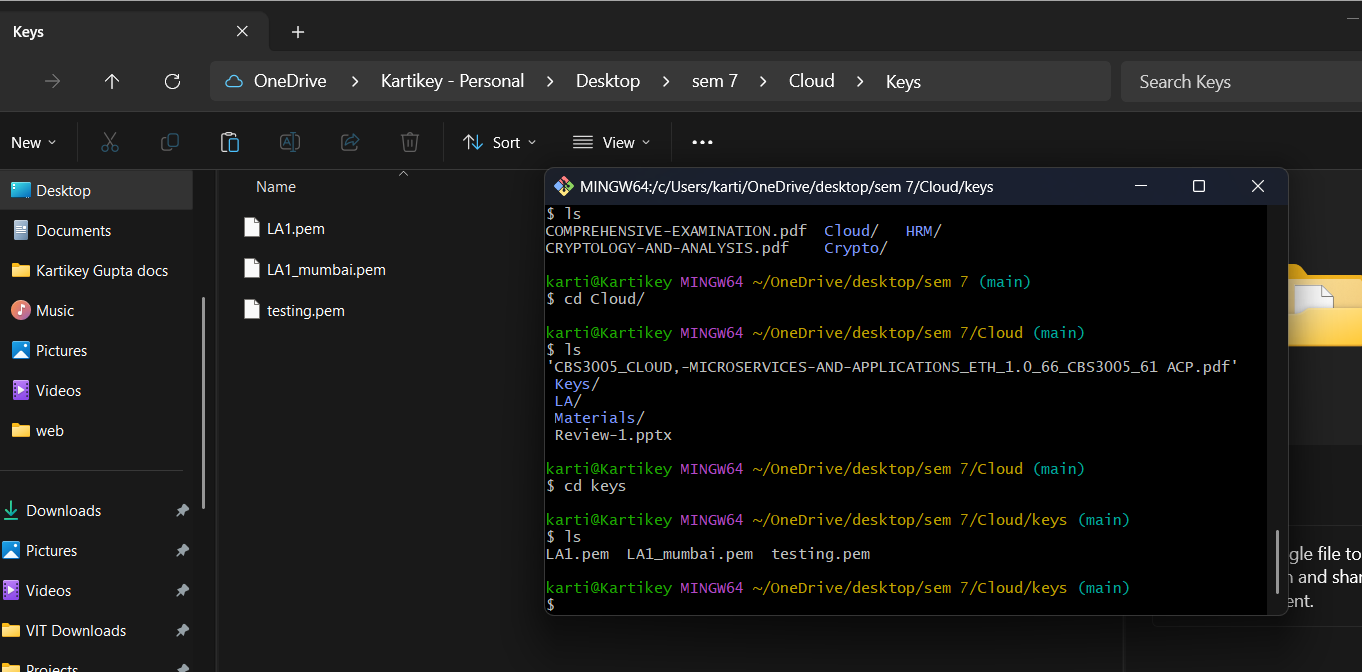
1. Security Group → Add rules:
   * SSH (22) → My IP only
   * MySQL (3306) → (only if you need remote DB access, otherwise skip).



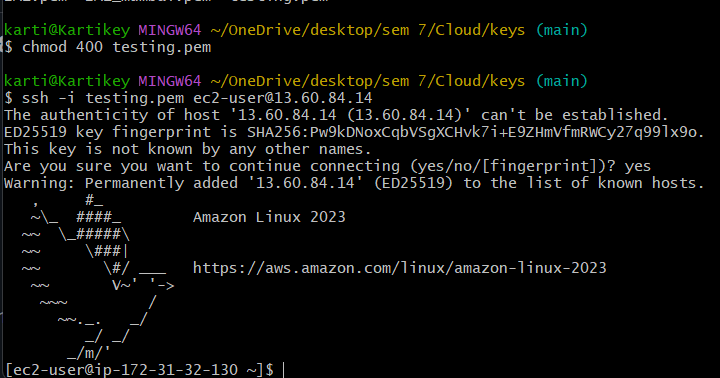
1. Click **Launch Instance**.

**2. Connect to EC2**

1. Download the .pem key (from step 4).



1. In your terminal:
2. chmod 400 key.pem
3. ssh -i key.pem ec2-user@<EC2-Public-IP>



(Replace <EC2-Public-IP> with your instance’s public IP).

**3. Install MySQL (MariaDB)**

Installing MySql

* sudo dnf update -y
* sudo dnf install mariadb105-server -y

Start and enable MySQL:

* sudo systemctl start mariadb
* sudo systemctl enable mariadb

**Log in to MySQL CLI**

**sudo mysql -u root -p**

CREATE DATABASE student;

USE student;

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50),

age INT

);

INSERT INTO students (name, age) VALUES ('Aman', 20), ('Priya', 21), ('Ravi', 22);

SELECT \* FROM students;

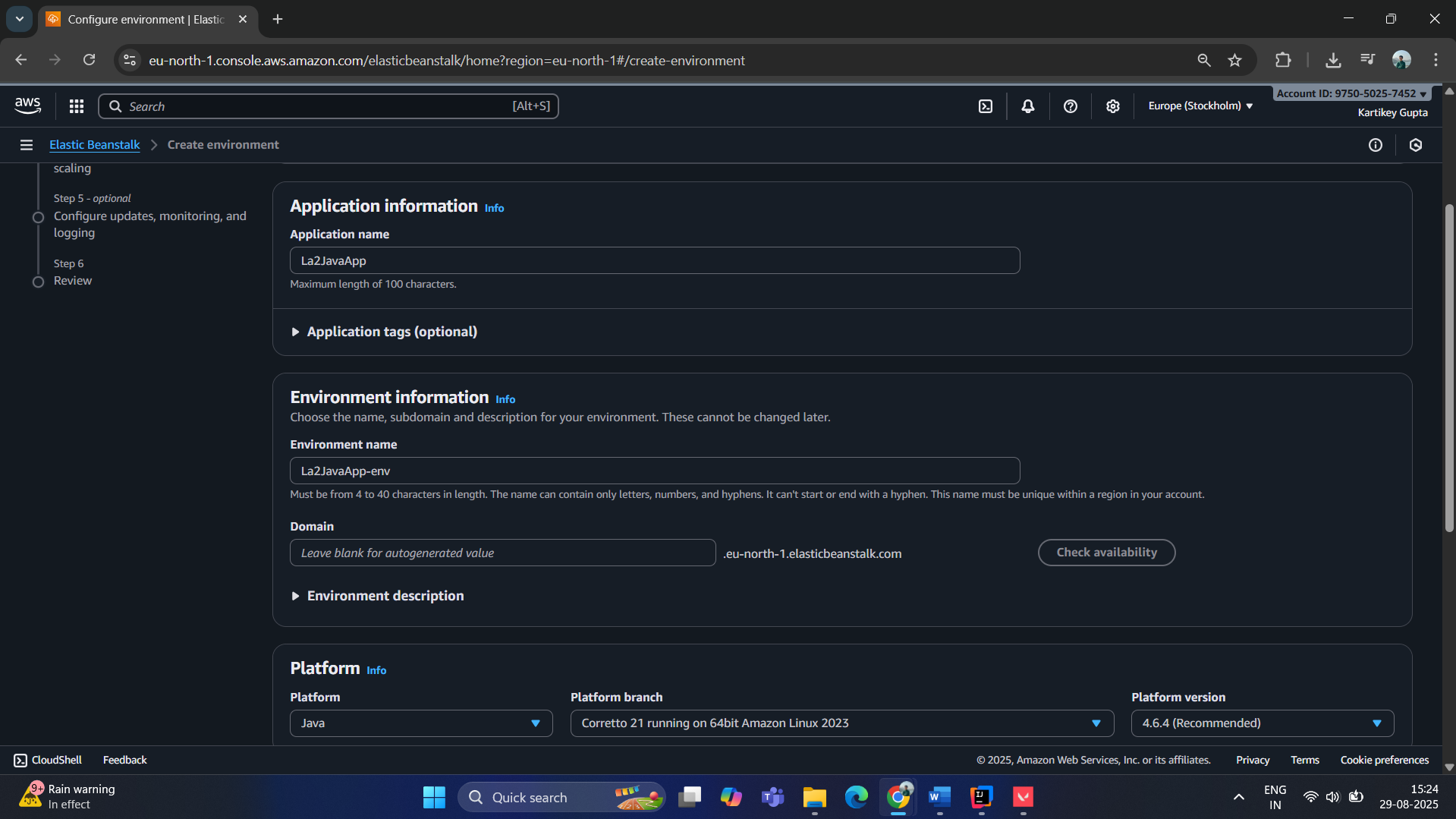
**Elastic Beanstalk**

**Step 2: Log in to AWS Management Console**

1. Go to AWS Console.
2. Log in with your AWS account.
3. Navigate to **Elastic Beanstalk**

**Step 3: Create a New Application**

1. Click **Create application**.
2. Enter **Application name**: La2JavaApp.
3. Add an optional description: Sample Spring Boot app.
4. Click **Create application**



**Step 5: Configure Environment and Platform**

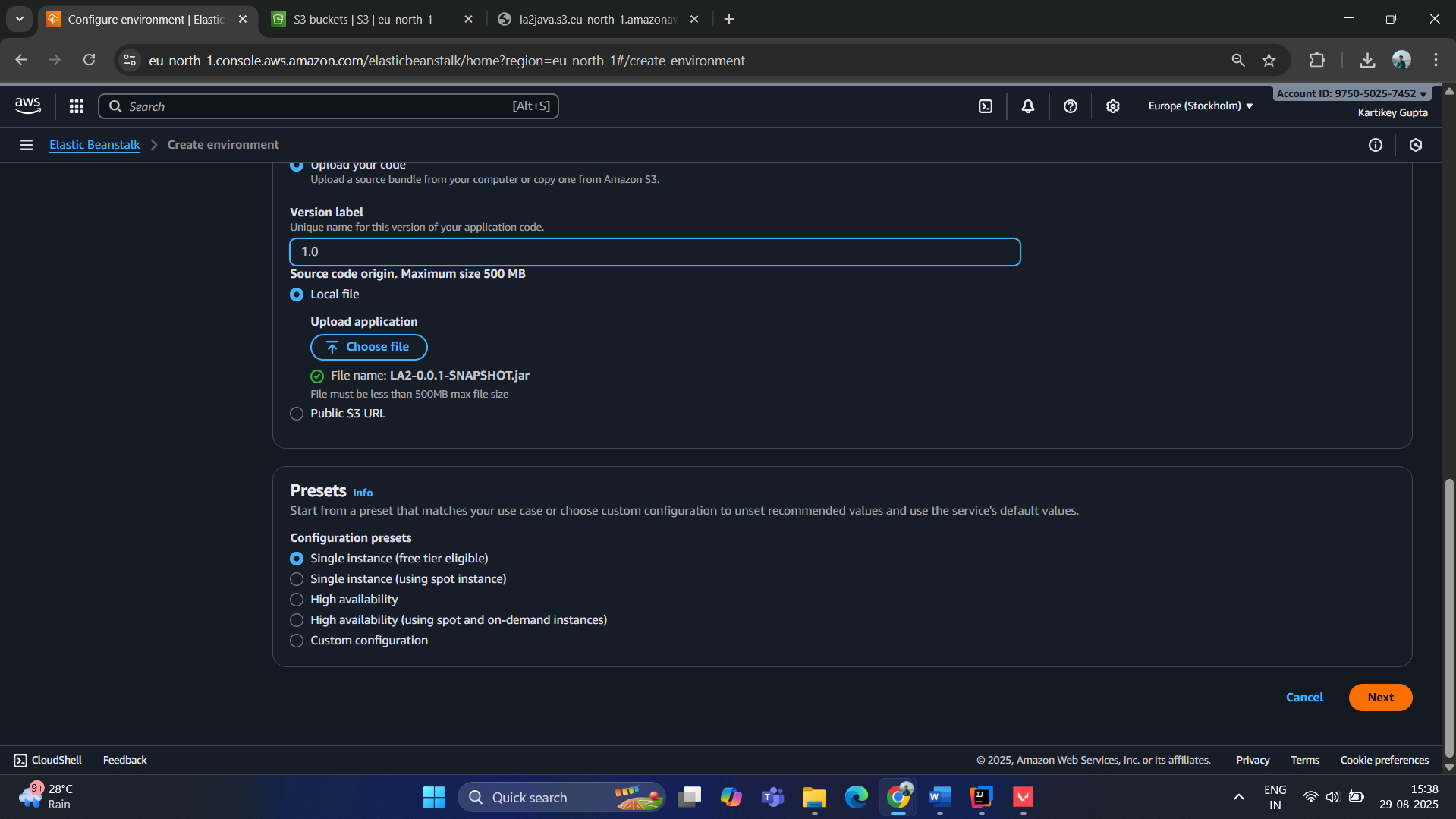
1. **Environment name**: La2JavaApp-env
2. **Platform**: Choose **Java**
   * For Spring Boot JAR: choose **Corretto 17** (or latest Java 17+)
3. **Platform branch**: Keep the default

**Step 6: Upload Your Application**

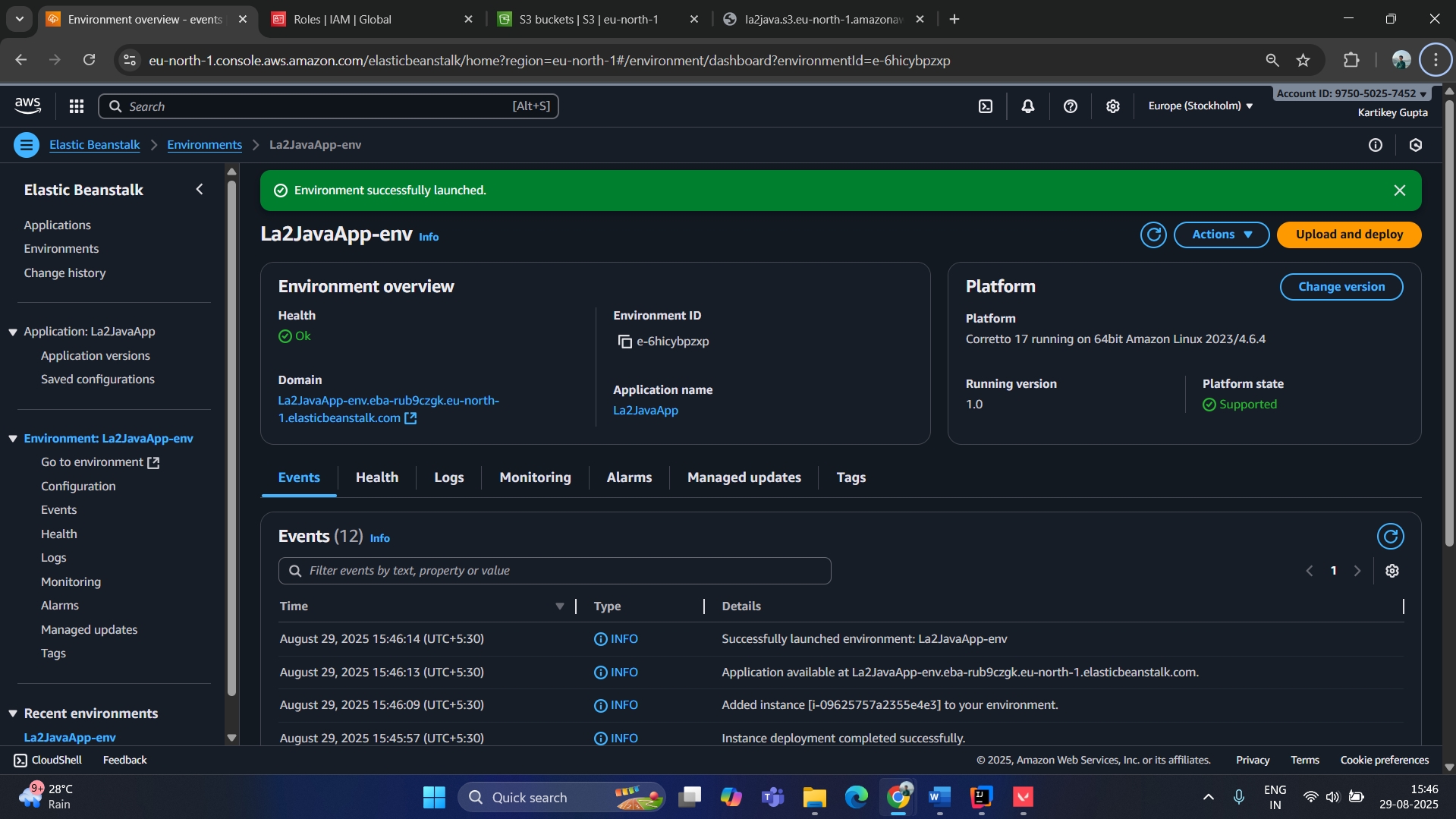
1. Under **Application code**, choose **Upload your code**.
2. Click **Choose file** and select your **JAR file** from target/.

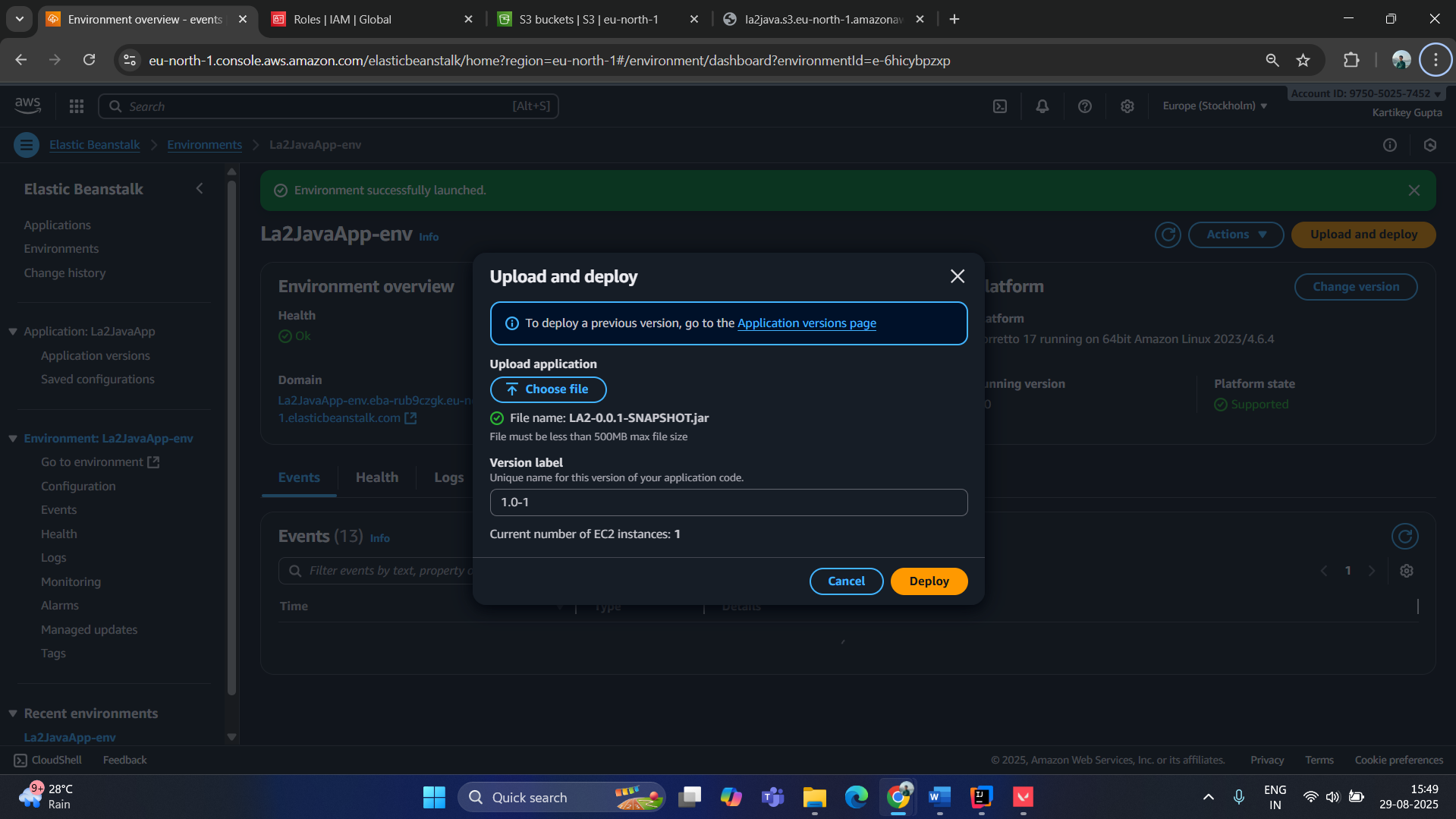
**Step 7: Optional Environment Settings**

1. **Instance type**: For testing, choose t2.micro (free tier eligible).
2. **Capacity**: Keep single instance.



Uploading JAR file:





**Lambda Python:**

import json

import urllib.parse

import boto3

print('Loading function')

s3 = boto3.client('s3')

def lambda\_handler(event, context):

# Uncomment for debugging

# print("Received event: " + json.dumps(event, indent=2))

# Get the object from the event

bucket = event['Records'][0]['s3']['bucket']['name']

key = urllib.parse.unquote\_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')

try:

response = s3.get\_object(Bucket=bucket, Key=key)

print("CONTENT TYPE: " + response['ContentType'])

return response['ContentType']

except Exception as e:

print(e)

print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))

raise e